Sure! Let's go through this PHP code step by step. It’s designed to handle **user session management**, **role-based access control**, **auto logout**, and **fine-grained permission checking**.

**🔐 1. Session Handling**

if (session\_status() == PHP\_SESSION\_NONE) {

session\_start();

}

* Checks if a session is already active using session\_status().
* If no session is active (PHP\_SESSION\_NONE), it starts a new one with session\_start().

**👤 2. Authentication Check**

if (!isset($\_SESSION['user\_id'])) {

header("Location:login.php");

exit;

}

* Verifies if the user is logged in by checking if user\_id exists in the session.
* If not, the user is redirected to the login page.

**🛡️ 3. Role-Based Access Function**

if (!function\_exists('checkRole')) {

function checkRole($requiredRole) {

if ($\_SESSION['role'] !== $requiredRole) {

echo "Access Denied!";

exit;

}

}

}

* Defines a reusable function checkRole($requiredRole) **only if it hasn't been defined already**.
* It checks if the user's role (from session) matches the required role.
* If the roles don't match, access is denied.

✅ Example:

checkRole('admin'); // Only admins can access this page

**⏱️ 4. Session Timeout (Auto Logout)**

if (isset($\_SESSION['last\_activity']) && (time() - $\_SESSION['last\_activity'] > 600)) {

session\_unset();

session\_destroy();

header("Location:login.php?message=Session expired");

exit;

}

$\_SESSION['last\_activity'] = time();

* Checks if the session has been inactive for **more than 10 minutes** (600 seconds).
* If yes:
  + The session is cleared and destroyed.
  + The user is redirected to the login page with a session-expired message.
* Regardless, the user's last\_activity is updated to the **current time** to reset the timer.

**🧾 5. Permission-Based Access Control**

function checkPermission($permission) {

include 'db\_connect.php';

$role = $\_SESSION['role'];

$query = "SELECT $permission FROM roles\_permissions WHERE role = ?";

$stmt = $conn->prepare($query);

$stmt->bind\_param("s", $role);

$stmt->execute();

$result = $stmt->get\_result();

$row = $result->fetch\_assoc();

if (!$row || !$row[$permission]) {

die("<script>alert('🚫 Access Denied! You do not have permission to view this page.'); window.location='dashboard.php';</script>");

}

}

**What this does:**

* Gets the user’s role from the session.
* Looks up the **permission** (e.g., can\_view\_news) for that role in the roles\_permissions table.
* If the permission is **not granted**, it shows an alert and redirects to the dashboard.

✅ Example usage:

checkPermission('can\_edit\_news');

**✅ Summary of What This Script Handles:**

| **Feature** | **Description** |
| --- | --- |
| ✅ Session check | Ensures user is logged in |
| ✅ Auto logout | Logs user out after 10 minutes of inactivity |
| ✅ Role-based restriction | Only allows access for specific roles |
| ✅ Permission check | Verifies access to specific features via database |

Want me to help you build the roles\_permissions table or example data for testing this system?